Vignette for *RTN*: reconstruction of transcriptional networks and analysis of master regulators.

Mauro AA Castro, Xin Wang, Michael NC Fletcher,
Florian Markowetz and Kerstin B Meyer *
kerstin.meyer@cancer.org.uk

October 17, 2016

Contents

*Cancer Research UK - Cambridge Research Institute, Robinson Way Cambridge, CB2 0RE, UK.

1 Overview

The package *RTN* is designed for reconstruction and analysis of transcriptional networks (TN) using mutual information [**?**]. It is implemented by S4 classes in *R* [**?**] and extends several methods previously validated for assessing transcriptional regulatory units, or regulons (*e.g.* MRA [**?**], GSEA [**?**], synergy and shadow [**?**]). The package computes the mutual information (MI) between annotated transcription factors (TFs) and all potential targets using gene expression data. It is tuned to deal with large gene expression datasets in order to build genome-wide transcriptional networks centered on TFs and regulons. Using a robust statistical pipeline, *RTN* allows user to set the stringency of the analysis in a stepwise process, including a boostrep routine designed to remove unstable associations. Parallel computing is available for critical steps demanding high-performance.

2 Quick start

2.1 Transcriptional network inference

- 1 - Load a sample dataset

  The `dt4rtn` dataset consists of a list with 6 objects used for demonstration purposes only. It was extracted, pre-processed and size-reduced from [**?**] and [**?**] and contains a named gene expression matrix (`gexp`), a data frame of with `gexp` annotation (`gexpIDs`), a named numeric vector with differential gene expression data (`pheno`), a data frame with `pheno` annotation (`phenoIDs`), a character vector with genes differentially expressed (`hits`), and a named vector with transcriptions factors (`tfs`).

```
> library(RTN)
> data(dt4rtn)
```

- 2 - Create a new TNI object and run pre-processing

  Objects of class *TNI* provide a series of methods to do transcriptional network inference from high-throughput gene expression data. In this 1st step, the generic function *tni.preprocess* is used to run several checks on the input data.

```
> #Input 1: 'gexp', a named gene expression matrix (samples on cols)
> #Input 2: 'transcriptionFactors', a named vector with TF ids (3 TFs for quick demonstration!)
> #Input 3: 'gexpIDs', an optional data frame with gene annotation (it can be used to remove duplicated genes)
> rtni <- new("TNI", gexp=dt4rtn$gexp,
+             transcriptionFactors=dt4rtn$tfs[c("PTTG1","E2F2","FOXM1")]
+             )
> rtni<-tni.preprocess(rtni,gexpIDs=dt4rtn$gexpIDs)
```

- 3 - Run permutation analysis

  The *tni.permutation* function takes the pre-processed *TNI* object and returns a transcriptional network inferred by mutual information (with multiple hypothesis testing corrections).

```
> rtni<-tni.permutation(rtni)
```

- 4 - Run bootstrap analysis

  In an additional step, unstable interactions can be removed by bootstrap analysis using the *tni.bootstrap* function, which creates a consensus bootstrap network (referred here as *refnet*).

```
> rtni<-tni.bootstrap(rtni)
```

- 5 - Run DPI filter

  In the TN each target can be linked to multiple TFs and regulation can occur as a result of both direct (TF-target) and indirect interactions (TF-TF-target). The Data Processing Inequality (DPI) algorithm [**?**] is used to remove the weakest interaction in any triangle of two TFs and a target gene, thus preserving the dominant TF-target pairs, resulting in the filtered transcriptional network (referred here as *tnet*). The filtered TN has less complexity and highlights the most significant interactions.

```
> rtni<-tni.dpi.filter(rtni)
```

- 6 - Get results

  All results available in the *TNI* object can be retrieved using the *tni.get* function:

```
> tni.get(rtni,what="summary")
> refnet<-tni.get(rtni,what="refnet")
> tnet<-tni.get(rtni,what="tnet")
```

- 7 - Build a graph

  The inferred transcriptional network can also be retrieved as an *igraph* [**?**] object using the *tni.graph* function. The graph object includes some basic network attributes pre-formatted for visualization in the R package *RedeR* [**?**].

```
> g<-tni.graph(rtni)
```

2.2 Transcriptional network analysis

- 1 - Create a new TNA object (and run TNI-to-TNA pre-processing)

  Objects of class *TNA* provide a series of methods to do enrichment analysis on transcriptional networks. In this 1st step, the generic function *tni2tna.preprocess* is used to convert the pre-processed *TNI* object to *TNA*, also running several checks on the input data.

```
> #Input 1: 'object', a TNI object with a pre-processed transcripional network
> #Input 2: 'phenotype', a named numeric vector of phenotypes
> #Input 3: 'hits', a character vector of gene ids considered as hits
> #Input 4: 'phenoIDs', an optional data frame with anottation used to aggregate genes in the phenotype
> rtna<-tni2tna.preprocess(object=rtni,
+                          phenotype=dt4rtn$pheno,
+                          hits=dt4rtn$hits,
+                          phenoIDs=dt4rtn$phenoIDs
+                          )
```

- 3 - Run MRA analysis pipeline

  The *tna.mra* function takes the *TNA* object and returns the results of the Master Regulator Analysis (RMA) [**?**] over a list of regulons from a transcriptional network (with multiple hypothesis testing corrections). The MRA computes the overlap between the transcriptional regulatory unities (regulons) and the input signature genes using the hypergeometric distribution (with multiple hypothesis testing corrections).

```
> rtna<-tna.mra(rtna)
```

- 4 - Run overlap analysis pipeline

  A simple overlap among all regulons can also be tested using the *tna.overlap* function:

```
> rtna<-tna.overlap(rtna)
```

- 5 - Run GSEA analysis pipeline

  Alternatively, the gene set enrichment analysis (GSEA) can be used to assess if a given transcriptional regulatory unit is enriched for genes that are differentially expressed among 2 classes of microarrays (*i.e.* a differentially expressed phenotype). The GSEA uses a rank-based scoring metric in order to test the association between gene sets and the ranked phenotypic difference. Here regulons are treated as gene sets, an extension of the GSEA statistics as previously described [**?**].

```
> rtna<-tna.gsea1(rtna)
```

- 6 - Get results

  All results available in the *TNA* object can be retrieved using the *tna.get* function:

```
> tna.get(rtna,what="summary")
> tna.get(rtna,what="mra")
> tna.get(rtna,what="overlap")
> tna.get(rtna,what="gsea1")
```

- 7 - Plot GSEA

  To visualize the GSEA distributions, the user can apply the *tna.plot.gsea1* function that plots the one-tailed GSEA results for individual regulons:

```
> tna.plot.gsea1(rtna, file="tna_test", width=6, height=4,
+                heightPanels=c(1,0.7,3),
+                ylimPanels=c(0,3.5,0,0.8))
```

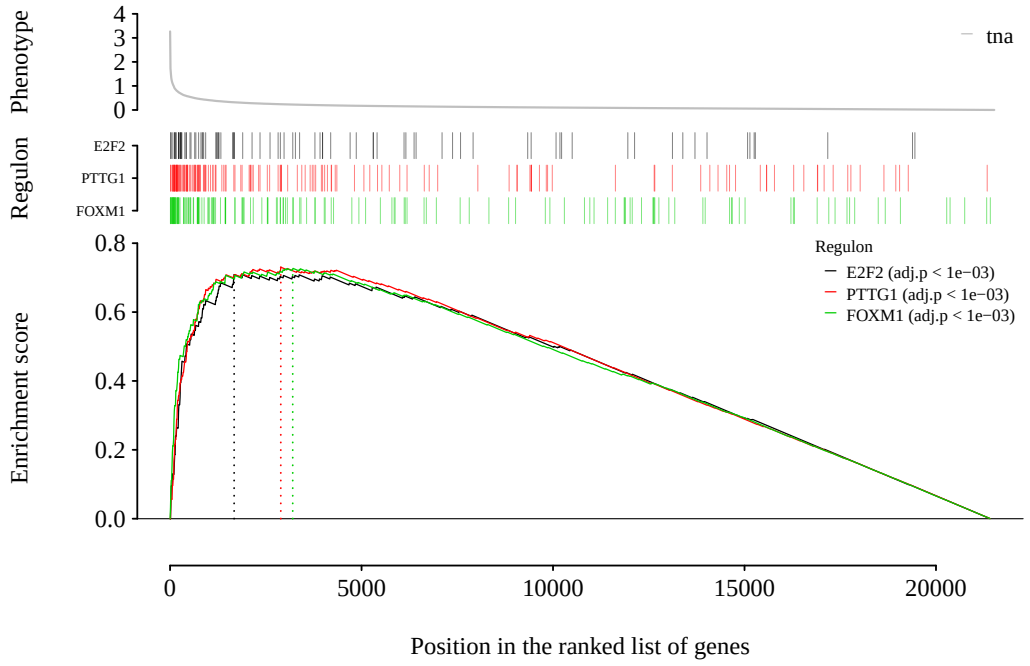

Figure 1: GSEA analysis showing genes in each regulon (as hits) ranked by their differential expression (as phenotype). This toy example illustrates the output from the *TNA* pipeline evaluated by the *tna.gsea1* method.

3 Session information

```
R version 3.3.1 (2016-06-21)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods
[7] base

other attached packages:
[1] RTN_1.12.0   igraph_1.0.1

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.7          magrittr_1.5         splines_3.3.1
 [4] BiocGenerics_0.20.0 MASS_7.3-45          IRanges_2.8.0
 [7] bit_1.1-12           lattice_0.20-34      minqa_1.2.4
[10] car_2.1-3            tools_3.3.1          nnet_7.3-12
[13] parallel_3.3.1       pbkrtest_0.4-6       grid_3.3.1
[16] nlme_3.1-128         data.table_1.9.6     mgcv_1.8-15
[19] ff_2.2-13            quantreg_5.29        snow_0.4-2
[22] MatrixModels_0.4-1  lme4_1.1-12          Matrix_1.2-7.1
[25] nloptr_1.0.4         RedeR_1.22.0         S4Vectors_0.12.0
[28] bitops_1.0-6         RCurl_1.95-4.8       limma_3.30.0
[31] pvclust_2.0-0        minet_3.32.0         stats4_3.3.1
[34] XML_3.98-1.4         SparseM_1.72         chron_2.3-47
```